School of Computing

NWC

NETWORK PACKET ROUTING IN DATA CENTERS USING GREEDY ALGORITHM

SRM INSTITUTE OF SCIENCE & TECHNOLOGY
(Deemed to be University u/s 3 of UGC Act, 1956)

PROBLEM STATEMENT:

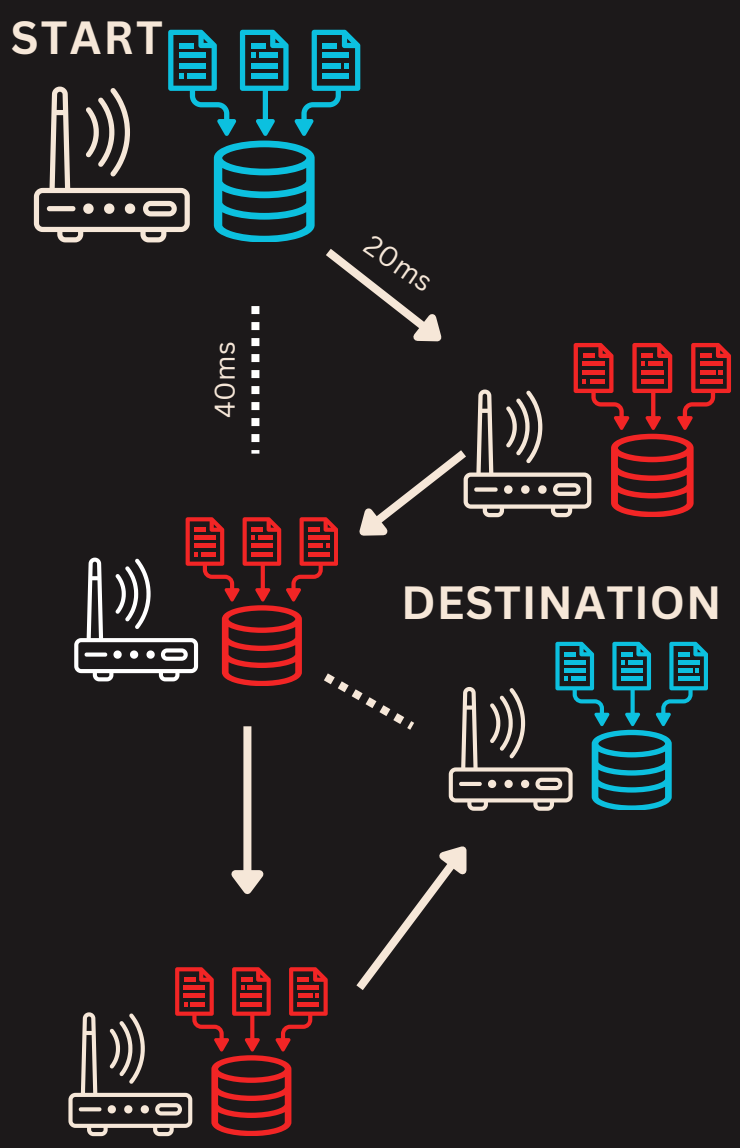
In large-scale data centers (like Google Cloud, AWS, or Microsoft Azure), millions of data packets are transmitted every second. The goal is to efficiently route packets from source to destination while minimizing latency and congestion.

LIMITATIONS

1. Genetic Algorithm:
Slow Convergence: Takes too long to find a solution, unsuitable for real-time routing.
High Computational Cost: Needs extensive iterations and population management.
Unpredictable Performance: The solution quality can vary significantly.

2. Dynamic Programming:
Memory Intensive: Stores multiple intermediate results, making it impractical for large networks.
High Time Complexity: Cannot handle dynamic, real-time changes efficiently.
Inflexibility: Requires the entire problem to be defined beforehand.

3. Graph-Based Methods (like Dijkstra/ A*):
Computationally Expensive: Finding shortest paths on large, real-time graphs takes too long.
Static Nature: Not adaptive to changes like congestion or link failures.
Complex Updates: Recalculating paths frequently is impractical in high-traffic environments.



TIME COMPLEXITY ANALYSIS

Step 1: Identify Recurrence Relation

1. Find the shortest latency neighbor → This takes O(N) time (scanning all neighbors).
2. Move the packet to the next router → O(1) time.
3. Repeat the process until the packet reaches the destination. In the worst case, we traverse all routers → O(N) steps.

Thus, the **recurrence relation** for the algorithm is:

$$T(N)=T(N-1)+O(N)$$

Step 2: Applying Master's Theorem

The standard Master's theorem applies to recurrences of the form:

$$T(N)=aT(N/b)+O(N^d)$$

Comparing with our recurrence:
a=1 (one recursive call)
b=1 (reducing problem size by 1 each step)
d=1 (O(N) work per step)

Since $d > \log_b a$, the complexity is determined by $O(N^d) = O(N)$.

Final Complexity :

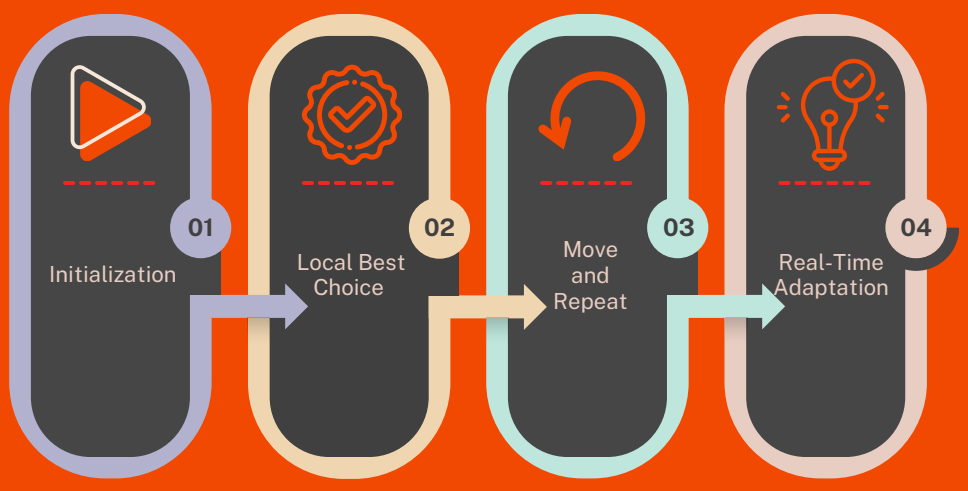
$$T(N) = O(N^2)$$

The **worst-case complexity is O(N²)** because, in a fully connected graph, scanning neighbors in each step leads to quadratic complexity.

SOLUTION

Fast Decision Making: Chooses the best local option at each step.
Low Computational Overhead: No need to precompute all possible paths.
Scalability: Can handle millions of packets simultaneously.

OPTIMIZED ALGORITHM



1. Represent the network as a graph with routers as nodes and links as edges (weighted by latency).
2. At each router, evaluate the neighboring nodes. Choose the next hop with the lowest latency and least congestion.
3. Move the packet to the selected next hop. Repeat the process until the packet reaches the destination.
4. Continuously update link weights based on current network congestion. Adjust the routing path dynamically if conditions change.

DOCUMENT

Dr. A. Prabhu Chakkaravarthy
21CSC204J
Design and Analysis of Algorithms
Mayukh Tilak (RA2311030010068)
Yashwanth Ch(RA2311030010087)
Anvita A(RA2311030010092)
Computer Science and Engineering(SC)-(Y1 Section)